



REST & Caching: Web Services, Accelerated

JAOO 2009

Stefan Tilkov, innoQ



<http://rest-http.info>



<http://soa-expertenwissen.de>



innoQ Deutschland GmbH **innoQ Schweiz GmbH**
Halskestraße 17 Gewerbestrasse 11
D-40880 Ratingen CH-6330 Cham
Phone +49 21 02 77 162-100 Phone +41 41 743 01 11
info@innoq.com · www.innoq.com

stefan.tilkov@innoq.com
**[http://www.innoq.com/blog/st/
@stilkov](http://www.innoq.com/blog/st/@stilkov)**

The REST Uniform Interface

identification
of resources

resource
manipulation
through
representations

hypermedia as
the engine of
application
state

self-descriptive
messages

The REST Uniform Interface

identification
of resources

resource
manipulation
through
representations

hypermedia as
the engine of
application
state

self-descriptive
messages

<http://example.com/orders?year=2008>

<http://example.com/customers/1234>

<http://example.com/orders/2007/10/776654>

<http://example.com/products/4554>

<http://example.com/processes/sal-increase-234>

The REST Uniform Interface

identification
of resources

resource
manipulation
through
representations

hypermedia as
the engine of
application
state

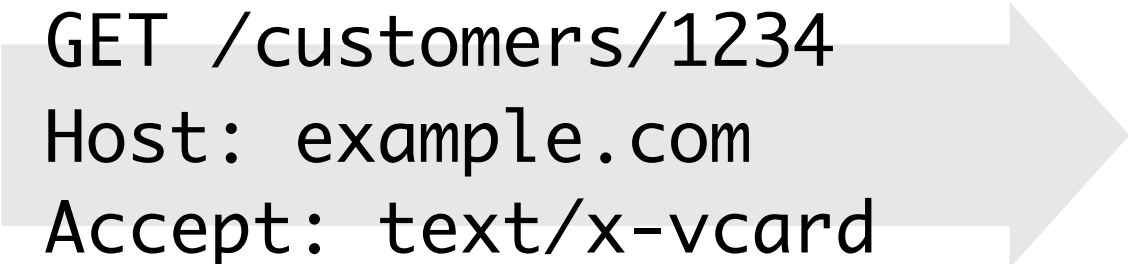
self-descriptive
messages



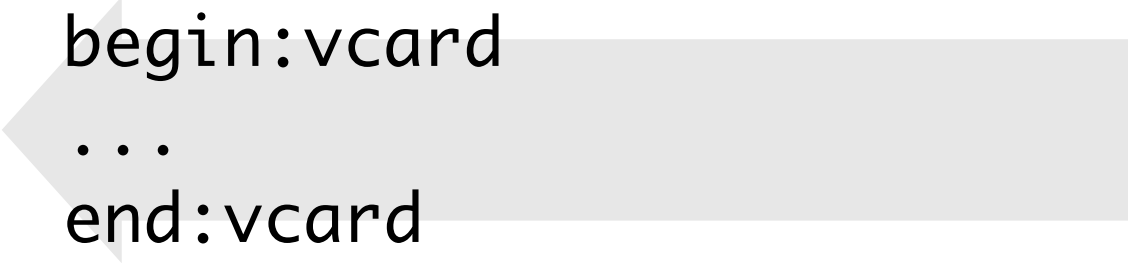
```
GET /customers/1234  
Host: example.com  
Accept: application/vnd.mycompany.customer+xml
```



```
<customer>...</customer>
```



```
GET /customers/1234  
Host: example.com  
Accept: text/x-vcard
```



```
begin:vcard  
...  
end:vcard
```

The REST Uniform Interface

identification
of resources

resource
manipulation
through
representations

hypermedia as
the engine of
application
state

self-descriptive
messages

```
<order self='http://example.com/orders/3321'>  
  <amount>23</amount>  
  <product ref='http://example.com/products/4554' />  
  <customer ref='http://example.com/customers/1234' />  
  <link rel='edit'  
    ref='http://example.com/order-edit/ACDB' />  
</order>
```

The REST Uniform Interface

identification
of resources

resource
manipulation
through
representations

hypermedia as
the engine of
application
state

self-descriptive
messages

Standard
Method

```
GET /service/customers/1234 HTTP 1.1
Host: www.example.com
User-Agent: XYZ 1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Keep-Alive: 300
Connection: keep-alive
If-Modified-Since: Fri, 02 Oct 2009 16:47:31 GMT
If-None-Match: "600028c-59fb-474f6852c9dab"
Cache-Control: max-age=60
```

Media Type

```
HTTP/1.1 200 OK
Date: Sun, 04 Oct 2009 16:56:25 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Fri, 02 Oct 2009 16:47:31 GMT
Etag: "600028c-59fb-474f6852c9dab"
Cache-Control: max-age=300
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 7160
Keep-Alive: timeout=15,max=91
Connection: Keep-Alive
Content-Type: application/xml
```

```
<?xml version='1.0' encoding='utf-8' ?>
```

...

visibility

Control Data

Data

getOrderDetails()

submitApplicationData()

updateQuote()

findMatchingBid()

initiateProcess()

cancelSubscription()

listAuctions()

getUsers()

getOrderDetails()

findMatchingBid()

GET

listAuctions()

getUsers()

initiateProcess()

submitApplicationData()

POST

PUT

updateQuote()

DELETE

cancelSubscription()

```
interface Resource {
    Resource(URI u)
    Response get()
    Response post(Request r)
    Response put(Request r)
    Response delete()
}
```

```
class CustomerCollection : Resource {
    ...
    Response post(Request r) {
        id = createCustomer(r)
        return new Response(201, r)
    }
    ...
}
```

generic

Any HTTP client
(Firefox, IE, curl, wget)

Any HTTP server

Caches

Proxies

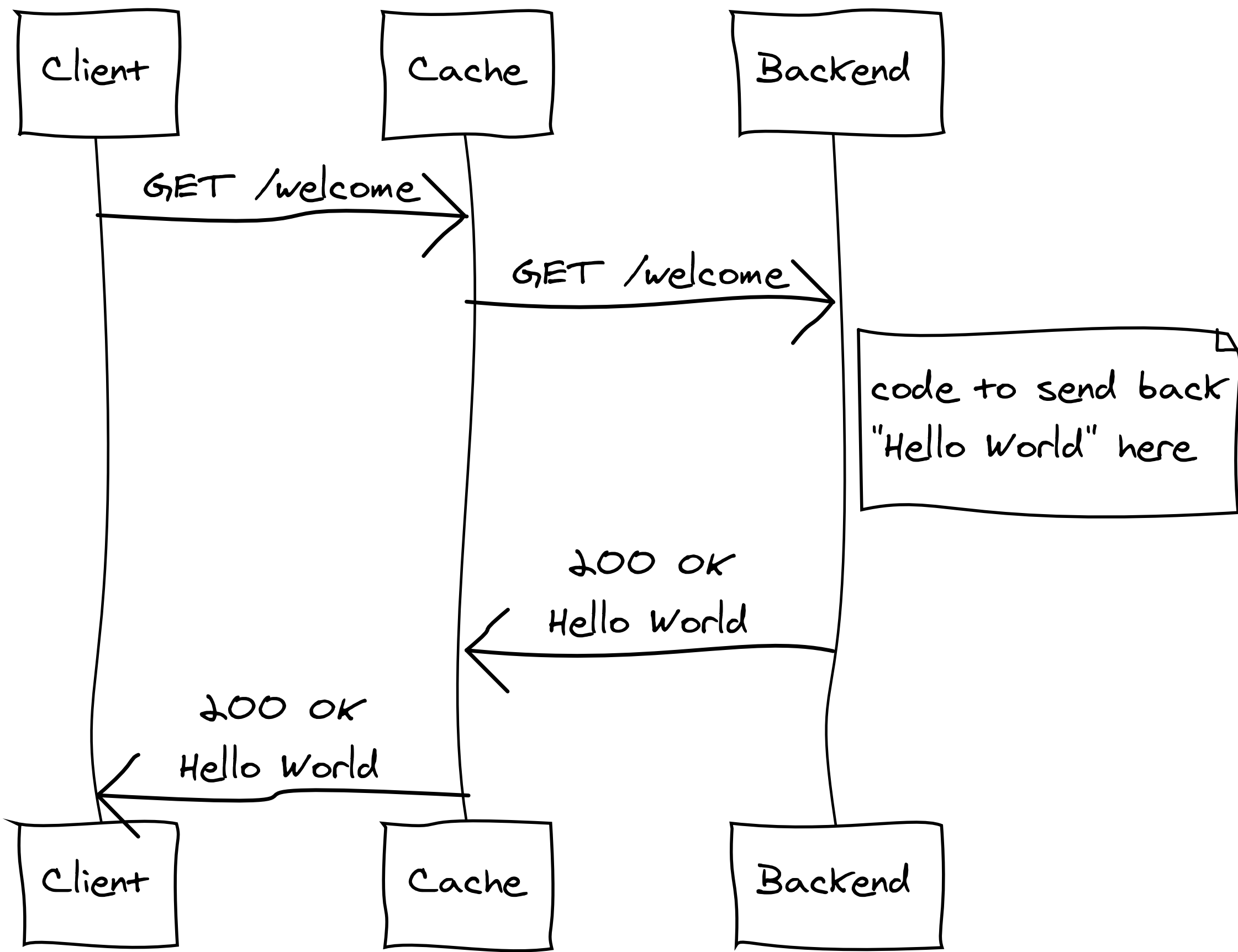
Google, Yahoo!, MSN

Anything that knows
your app

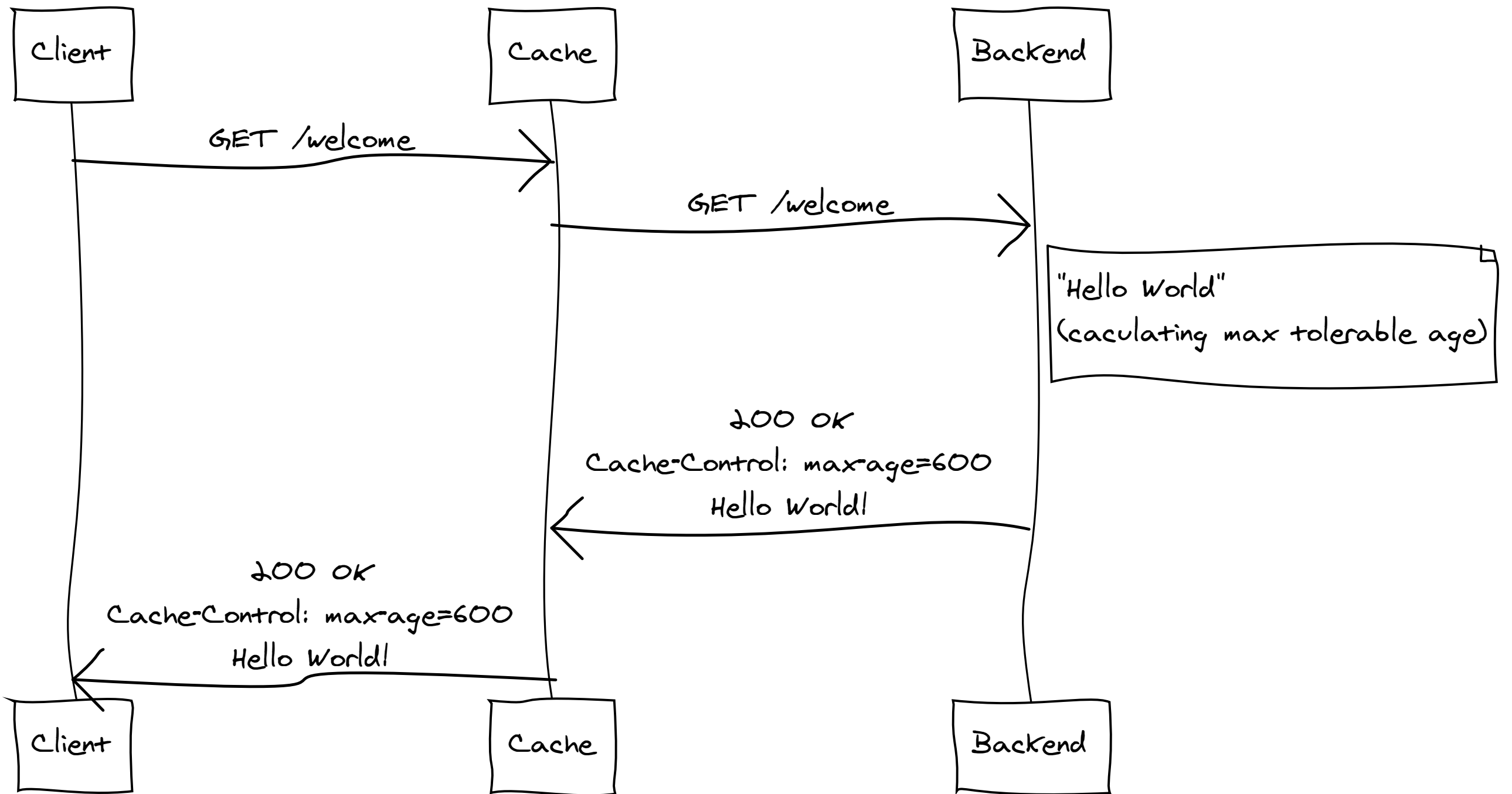
specific

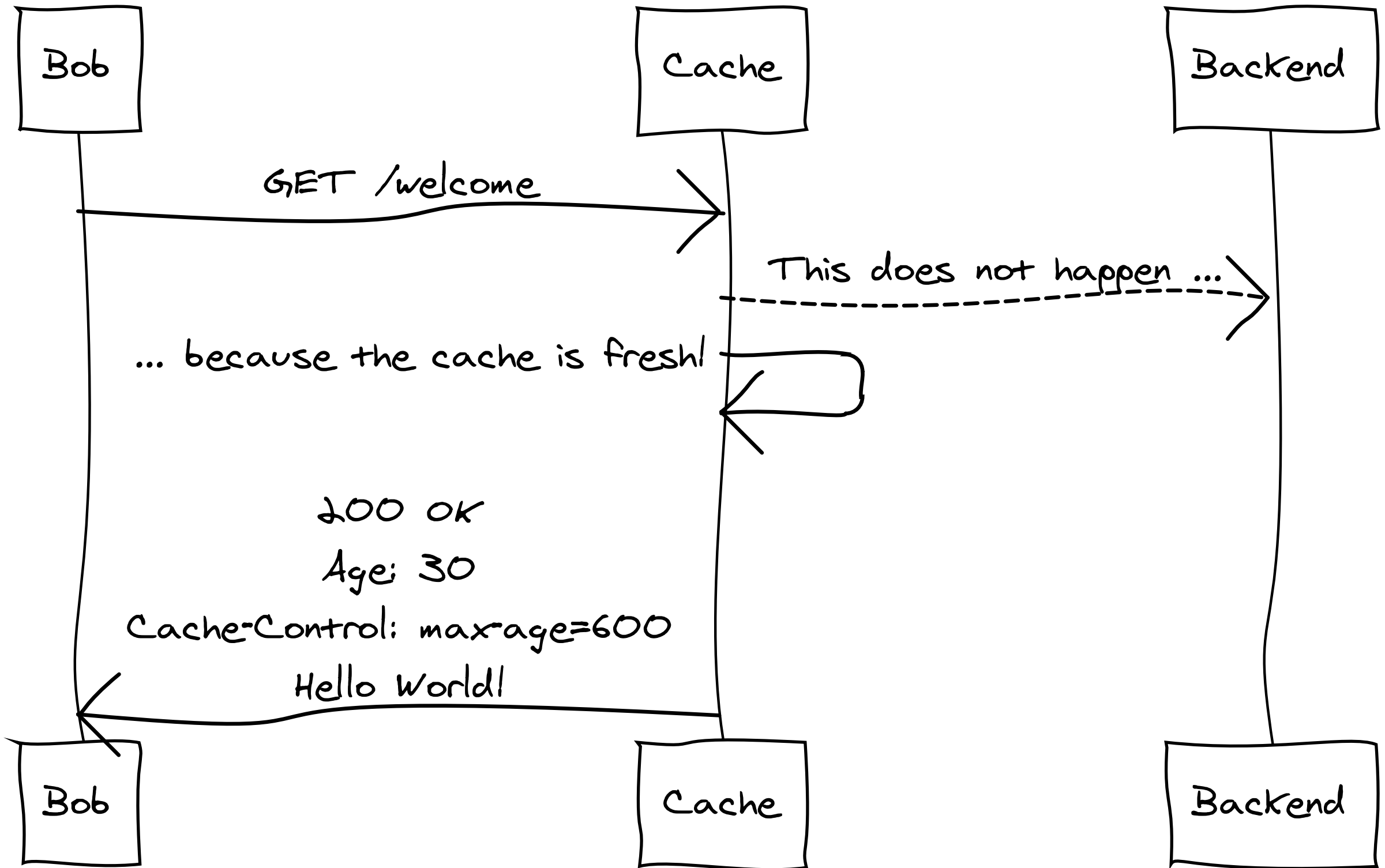
Caching Models

Note: Thanks to Ryan Tomayko for letting me steal-base some work on his diagrams from <http://tomayko.com/writings/things-caches-do>

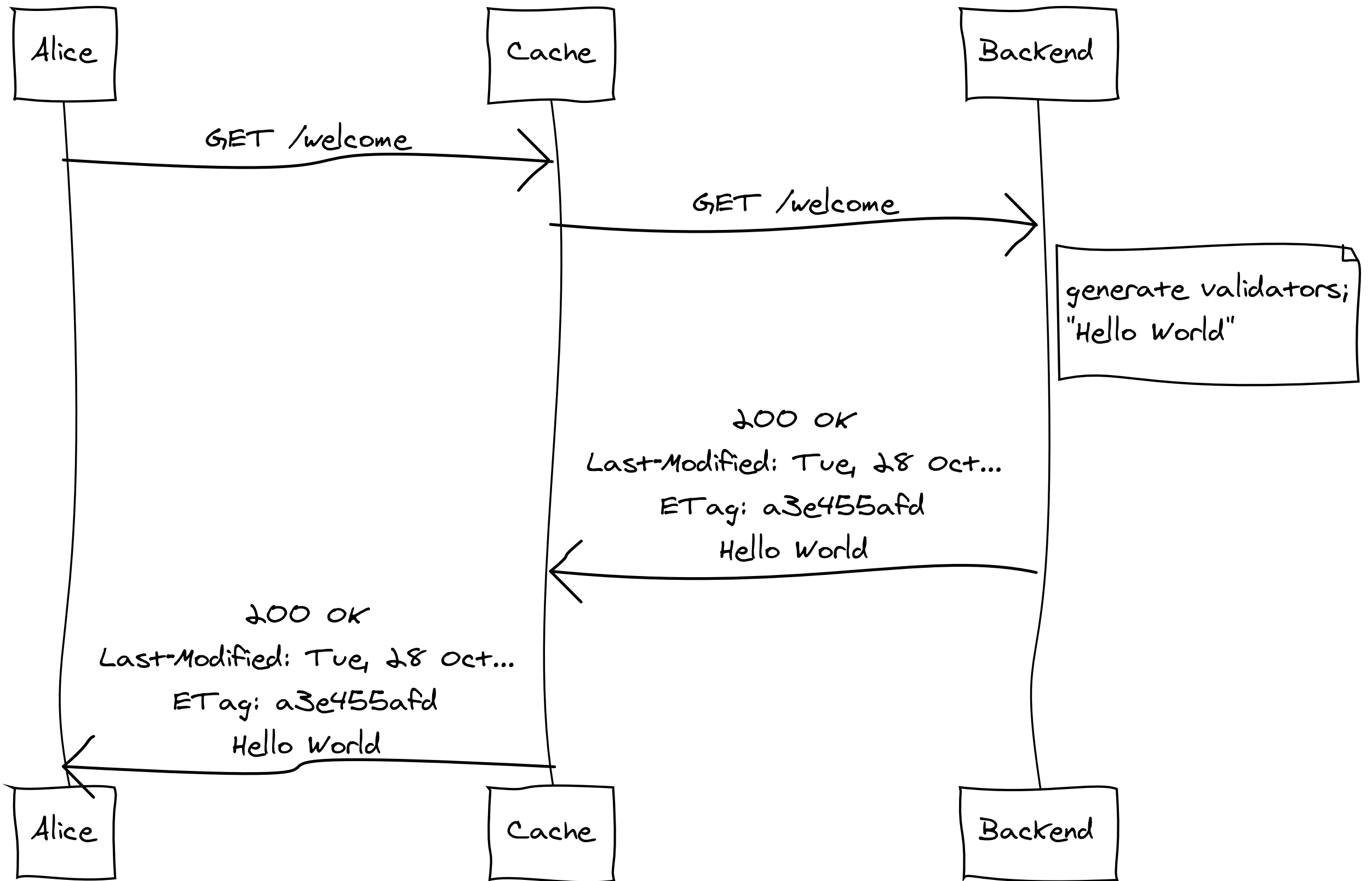


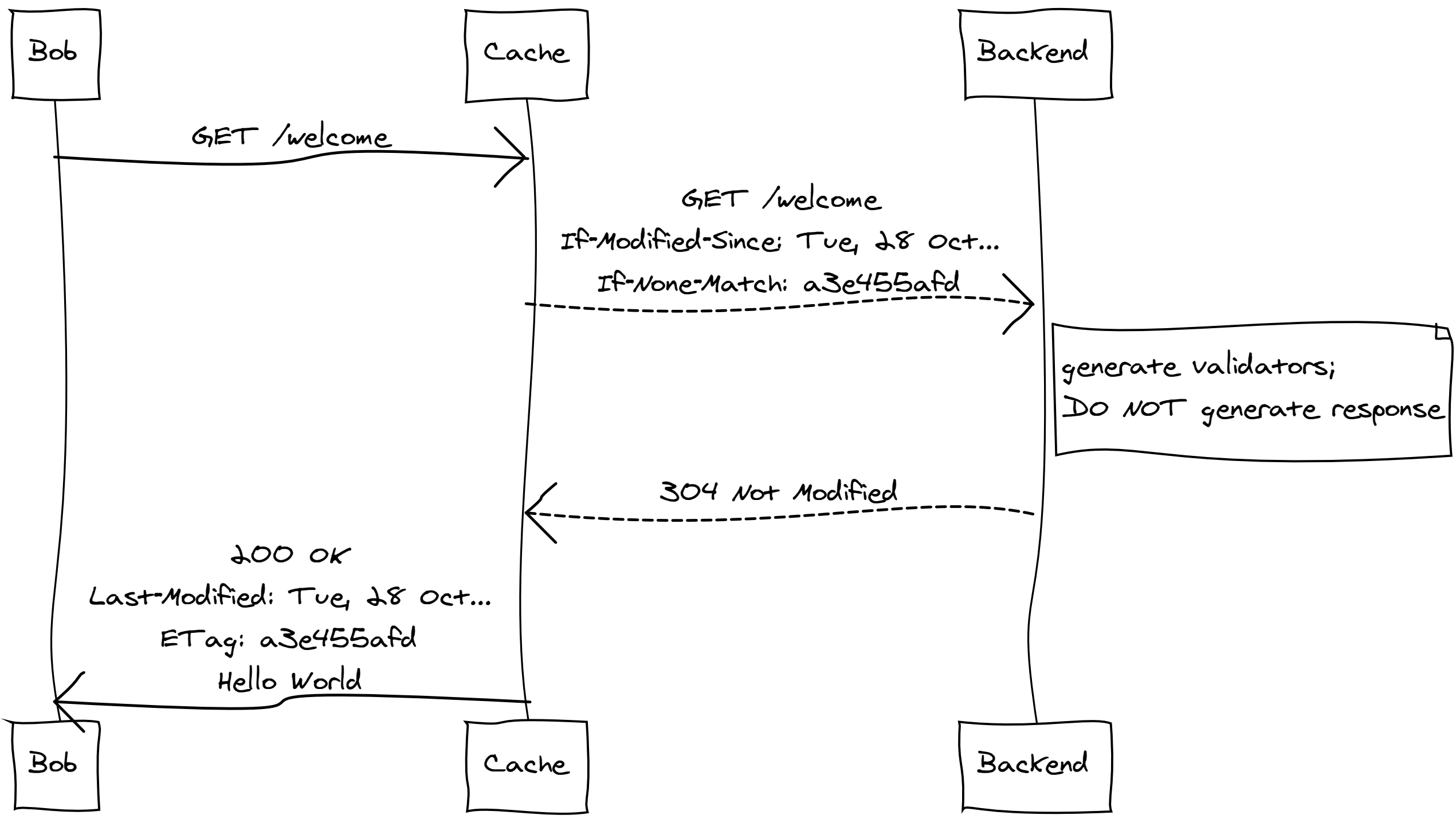
Expiration



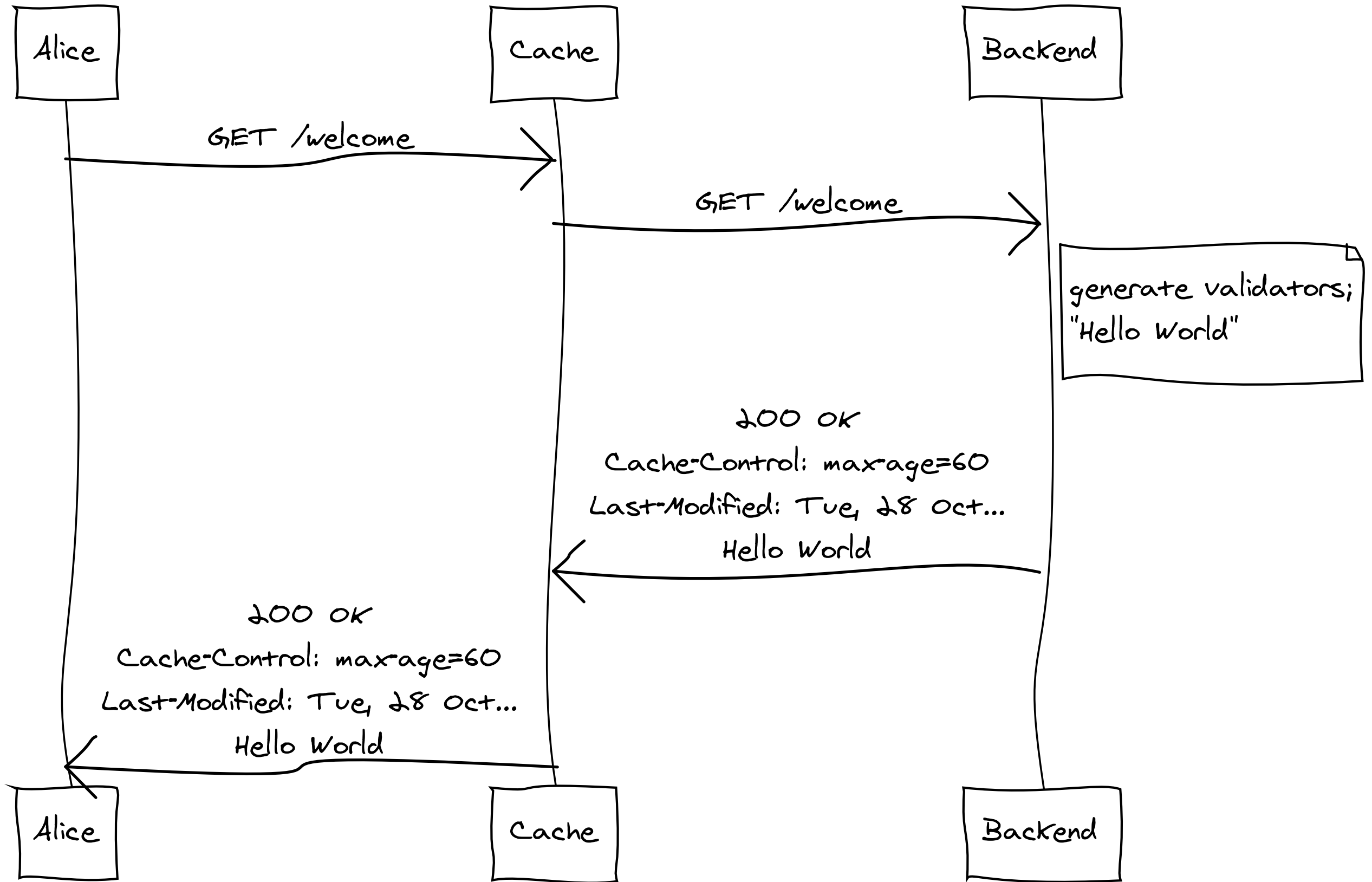


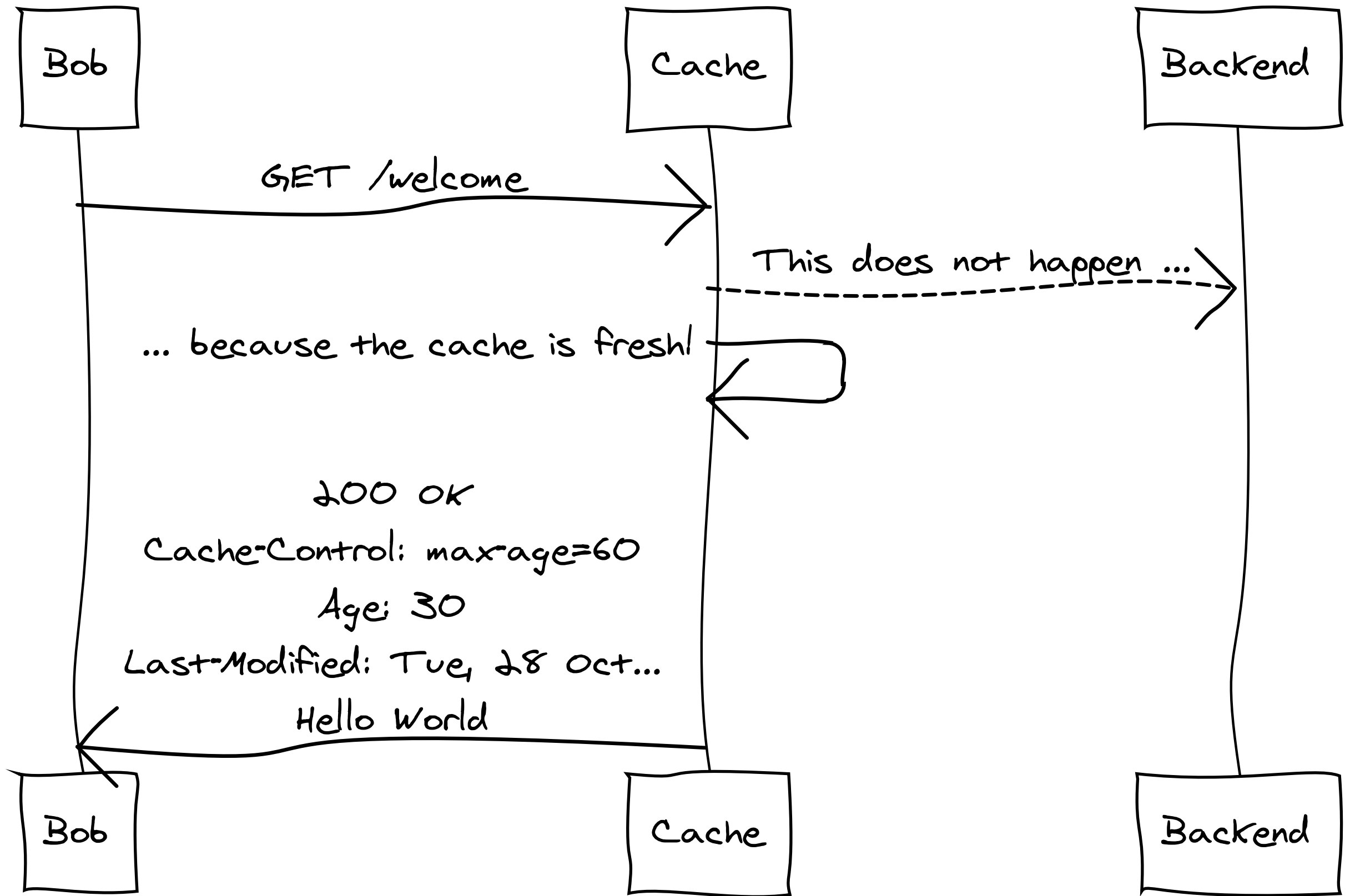
Validation

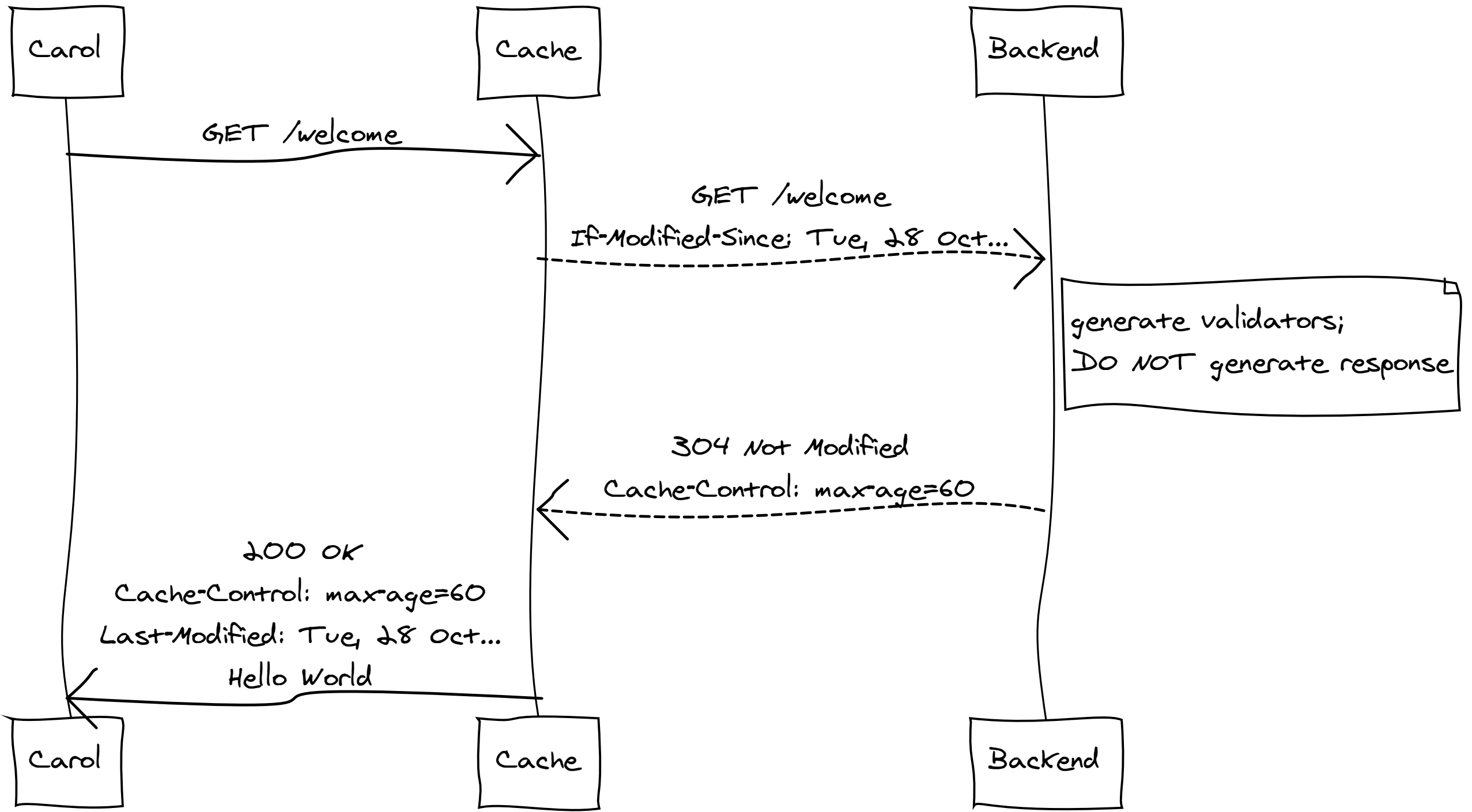




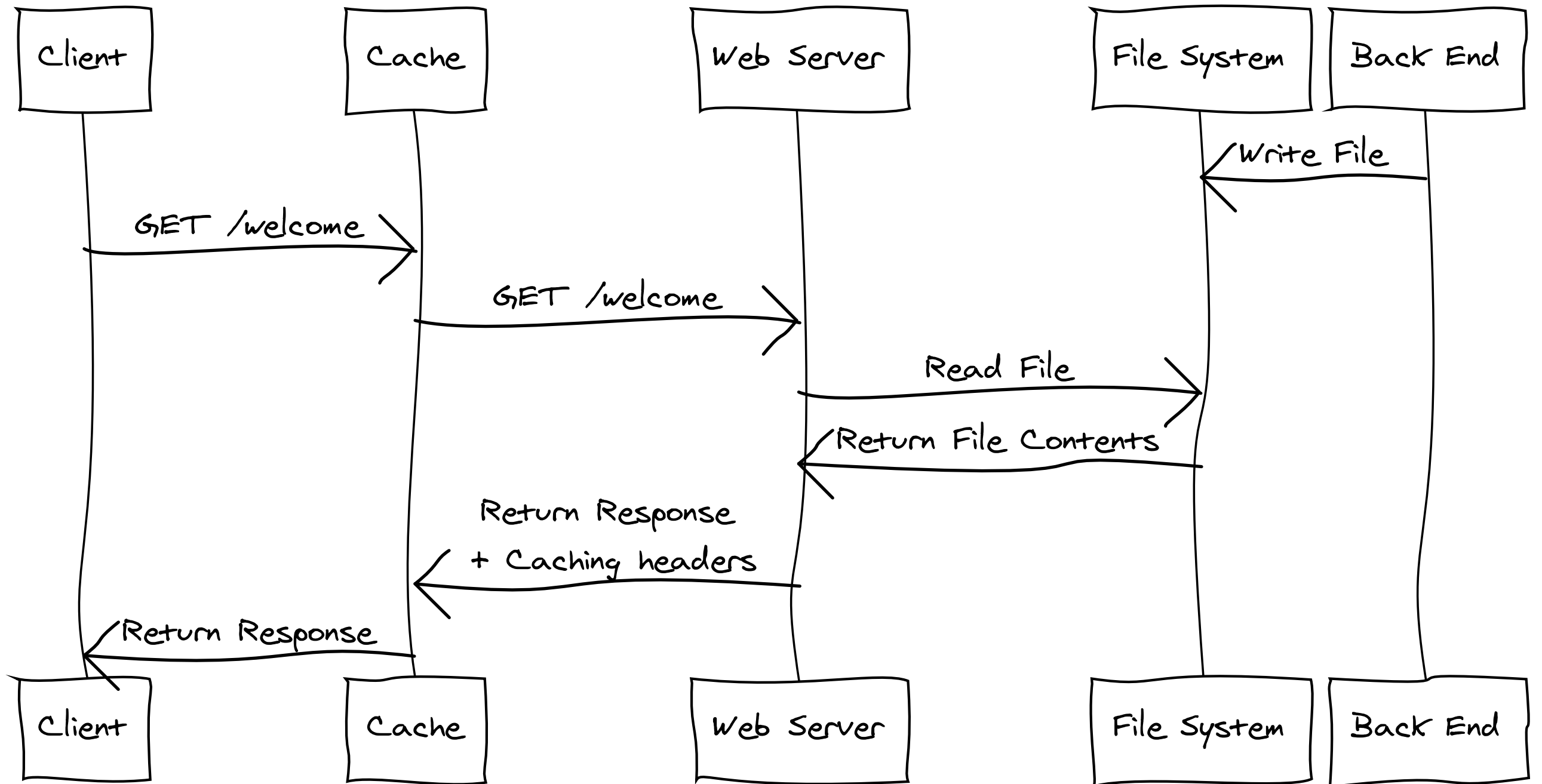
Combination







Implementation



Apache FileETag

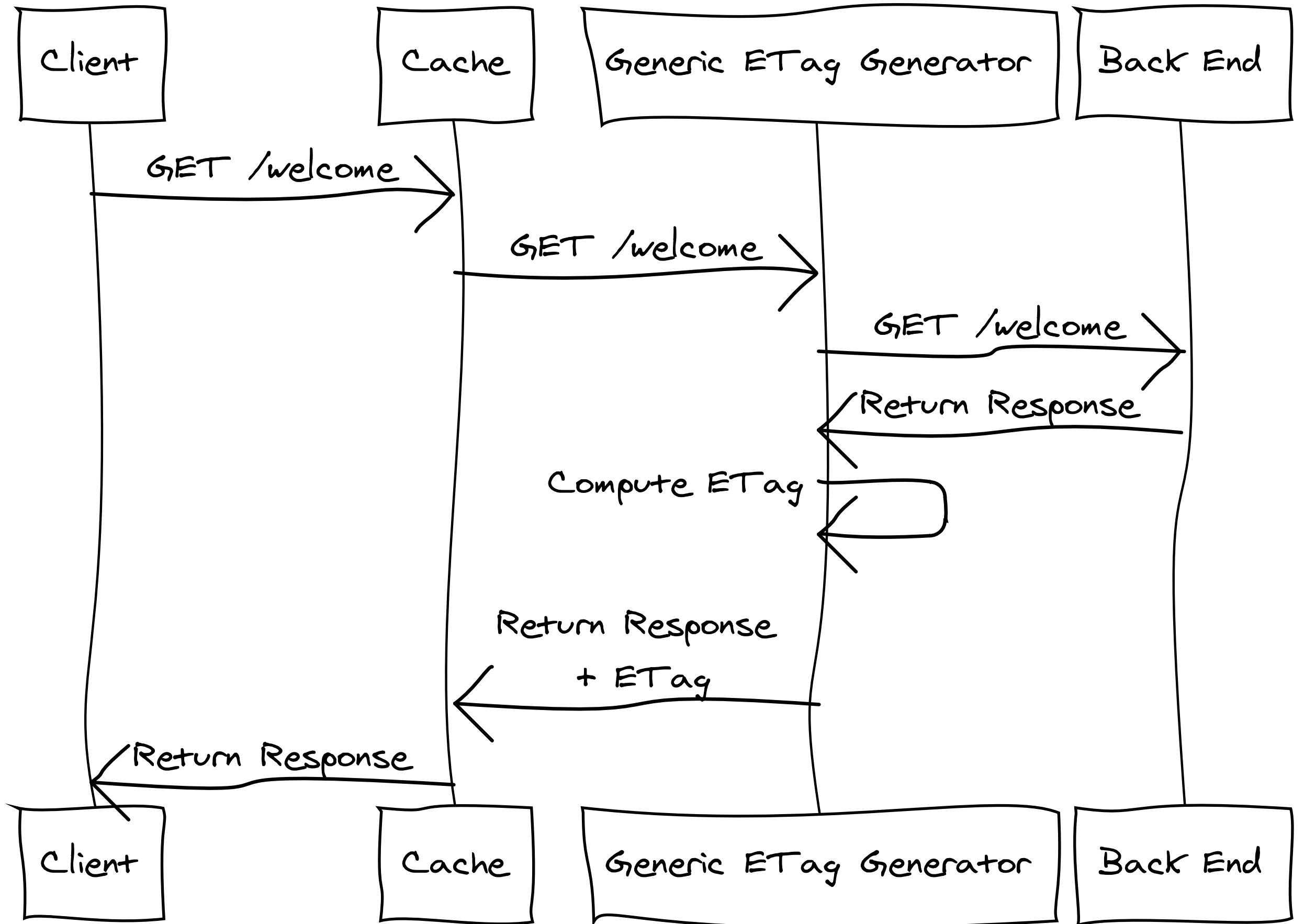
`FileETag Inode MTime Size`

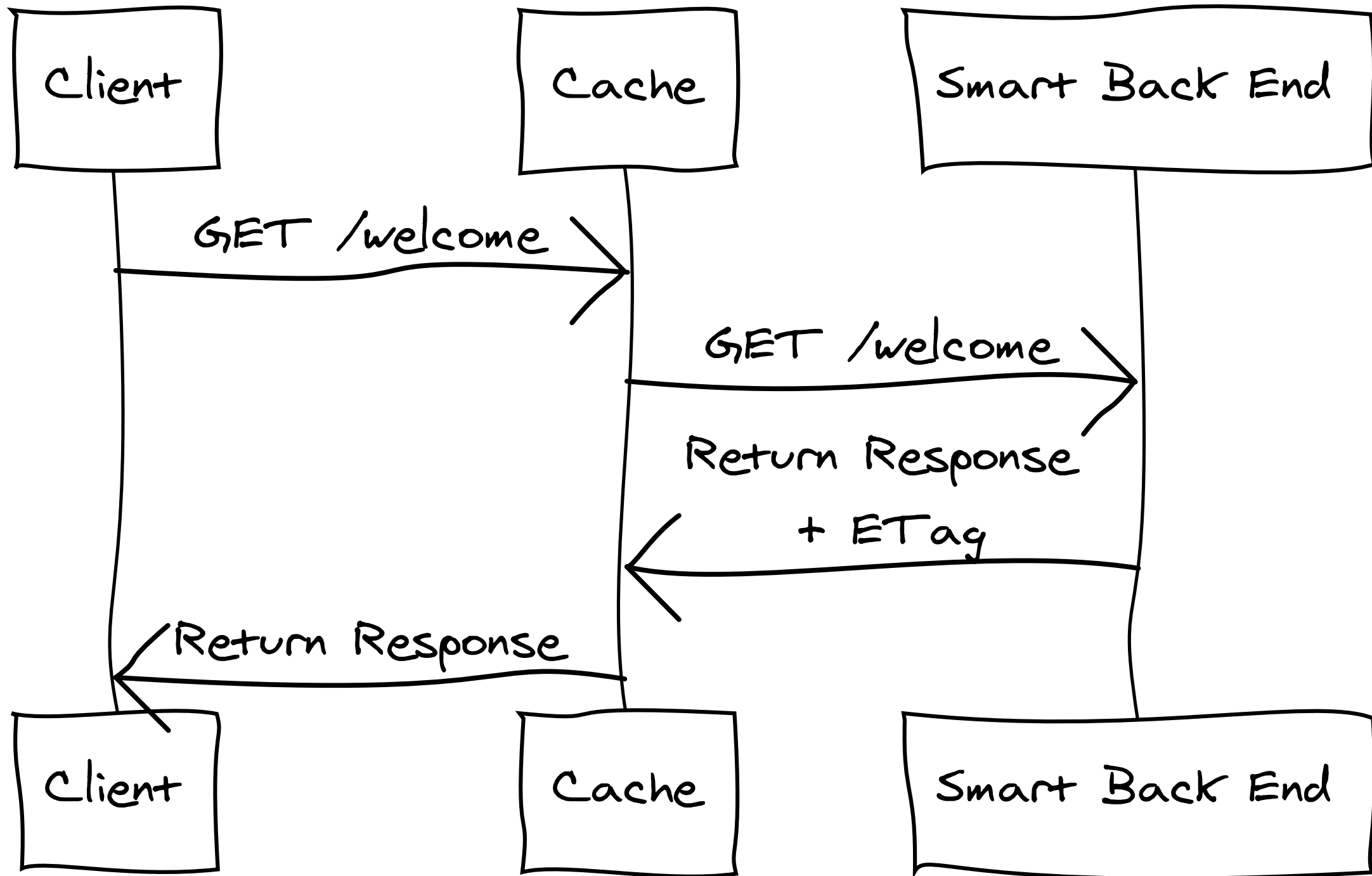
date and time the file
was last modified

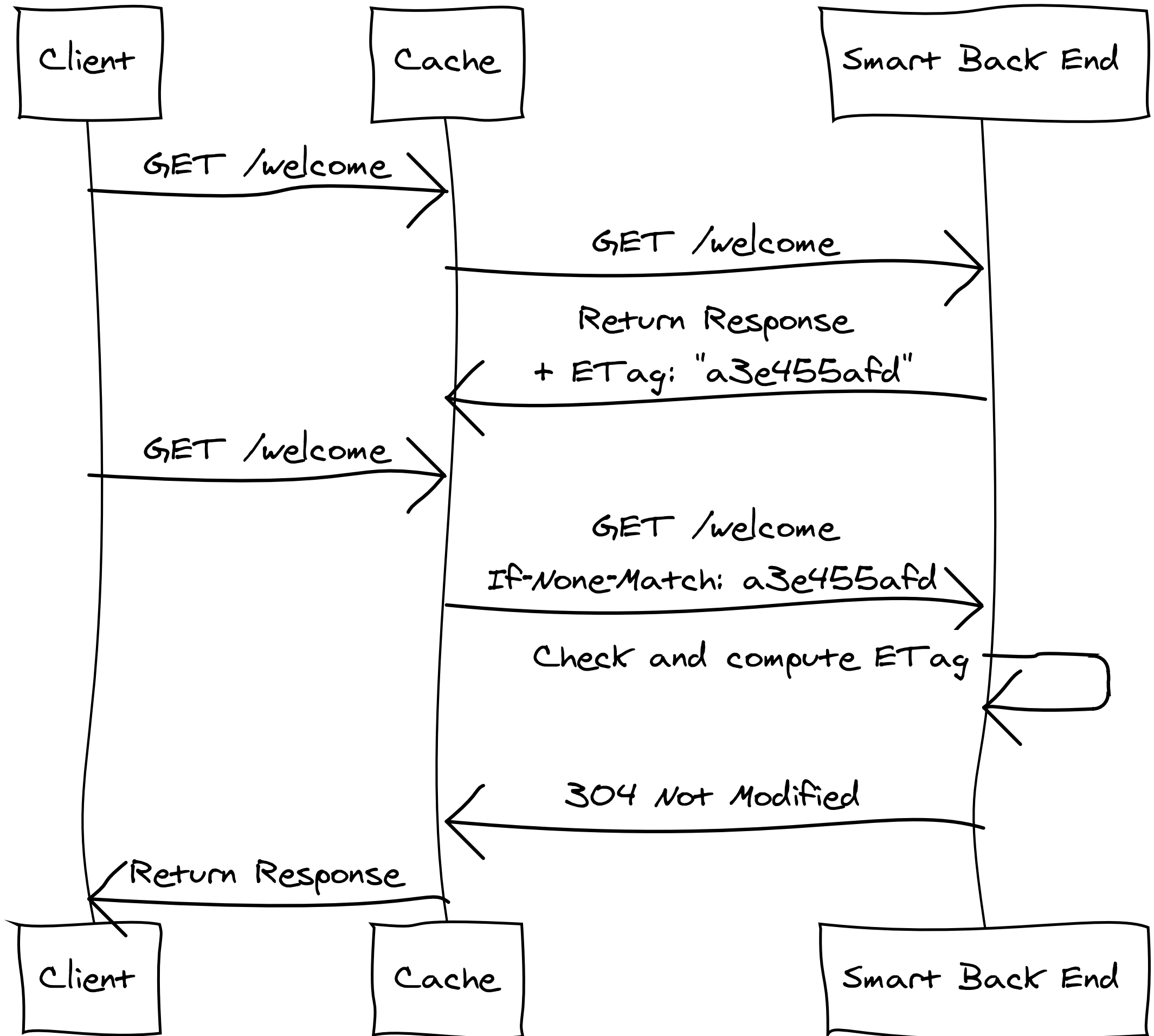
number of bytes
in the file

file's i-node number

ETag Depth



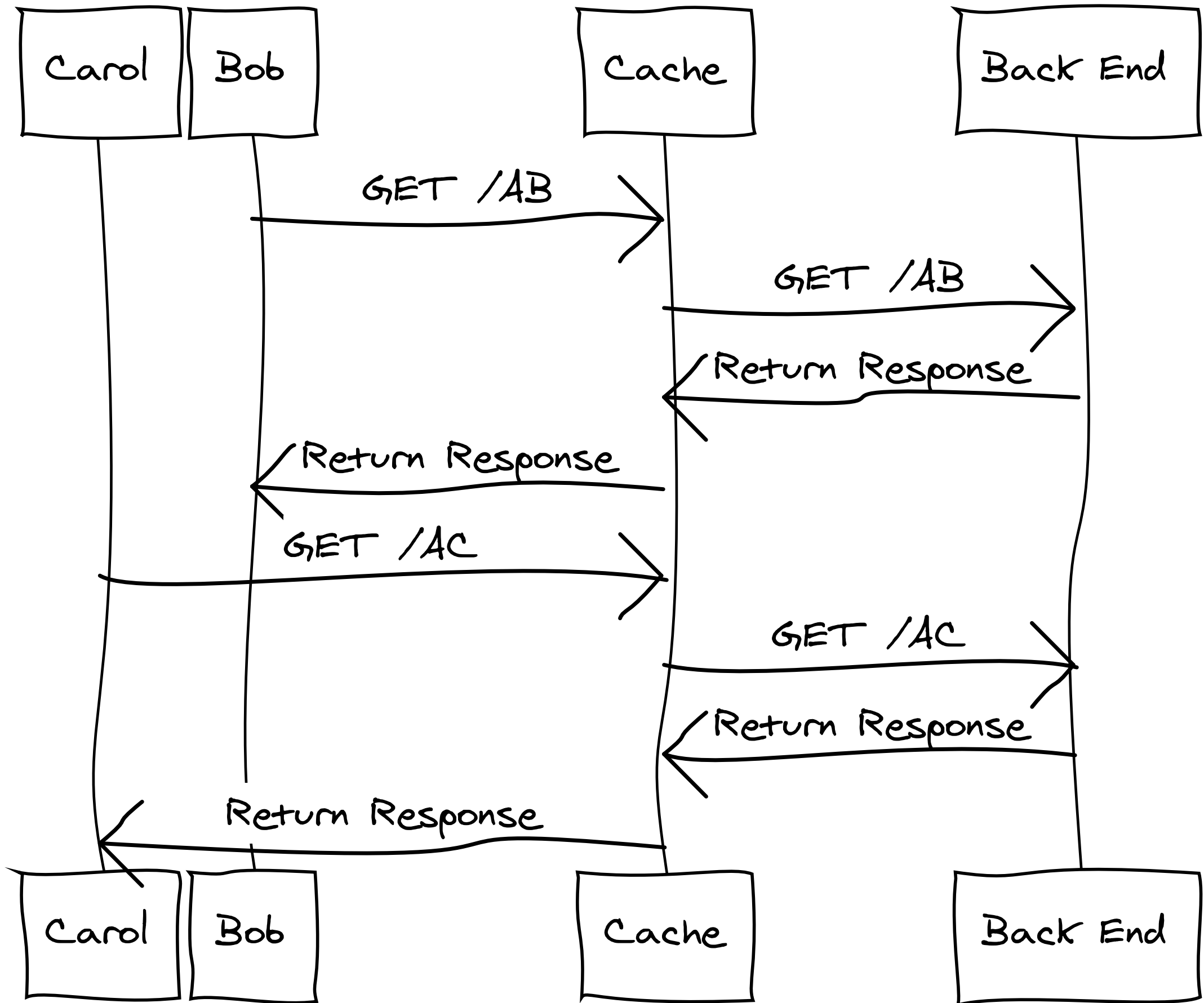


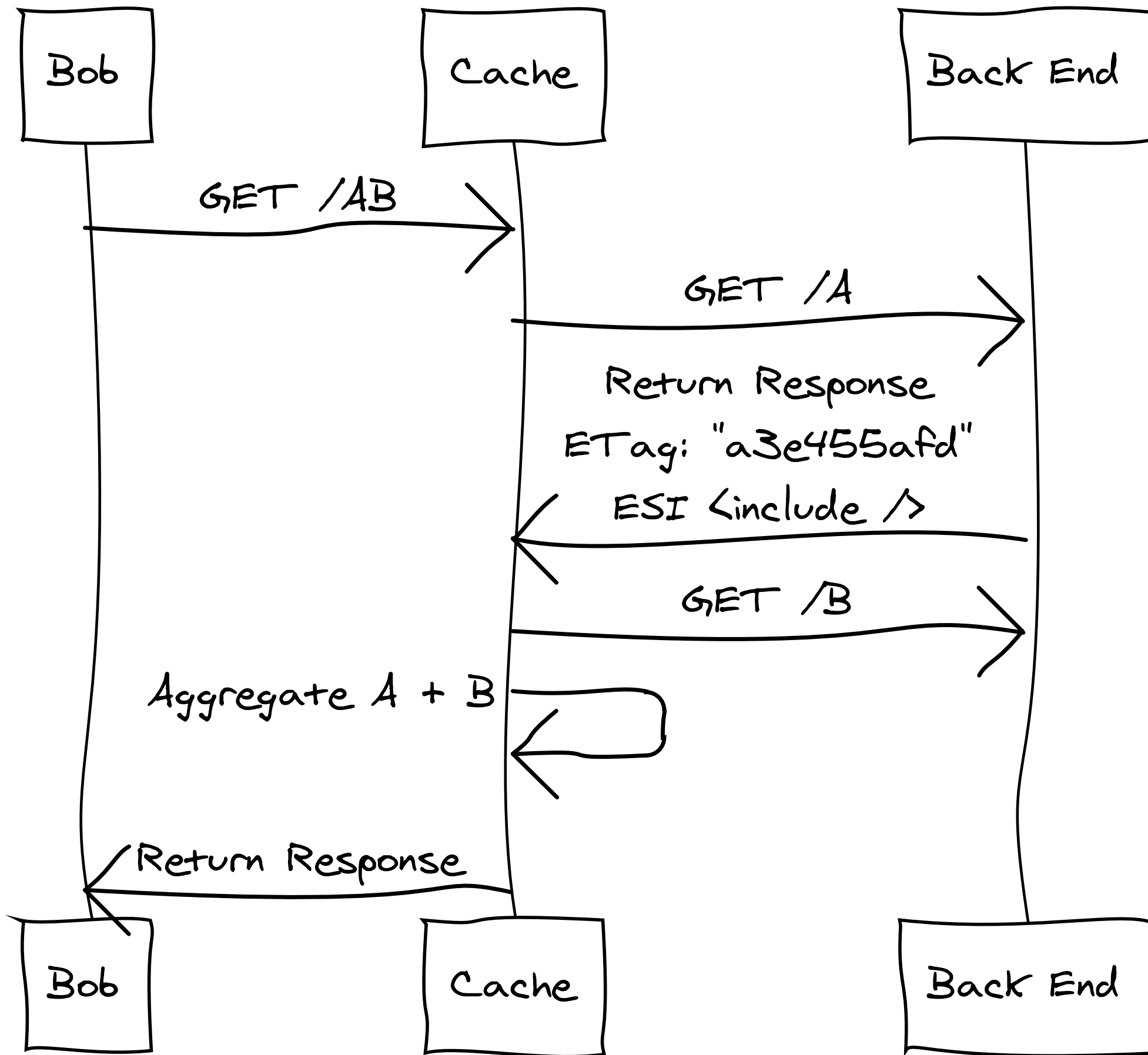


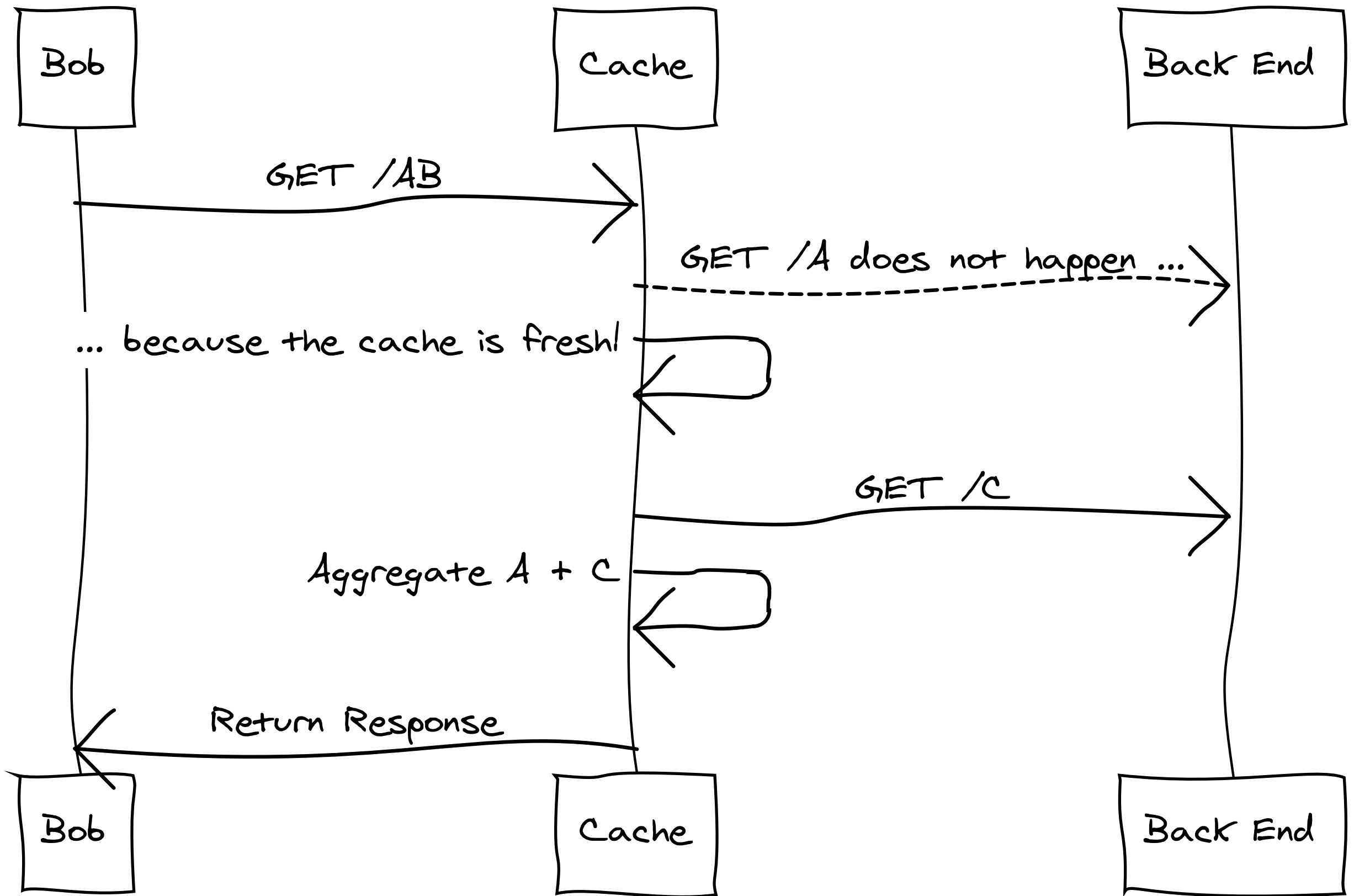
Edge Side Includes (ESI)

```
<esi:include  
  src="http://example.com/1.html"  
  alt="http://bak.example.com/2.html"  
  onerror="continue"/>
```

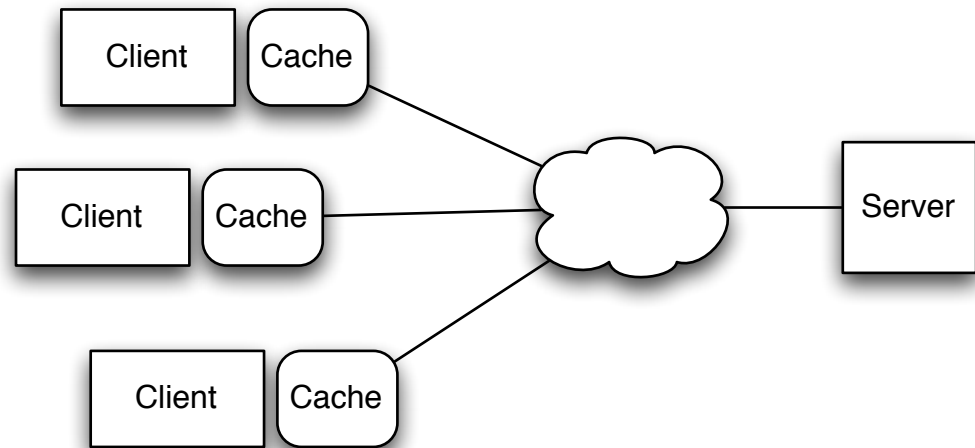
```
<esi:include  
  src="http://example.com/search?query=${QUERY_STRING{query}}"/>
```



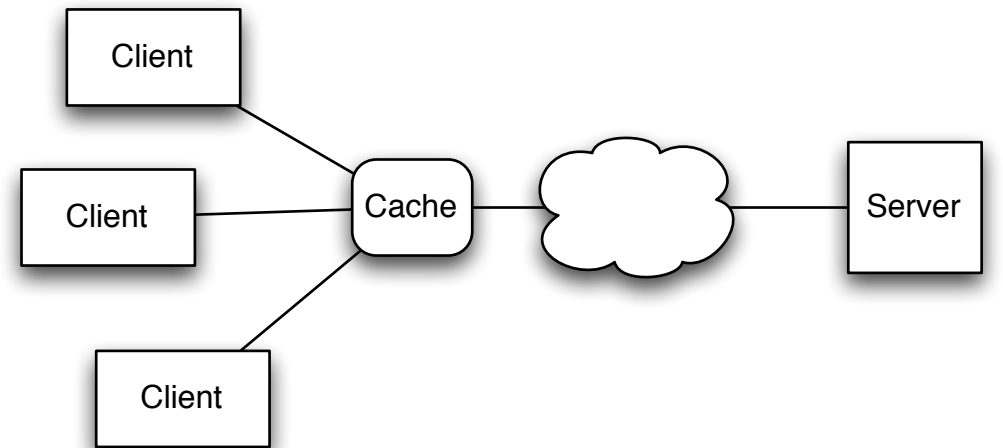




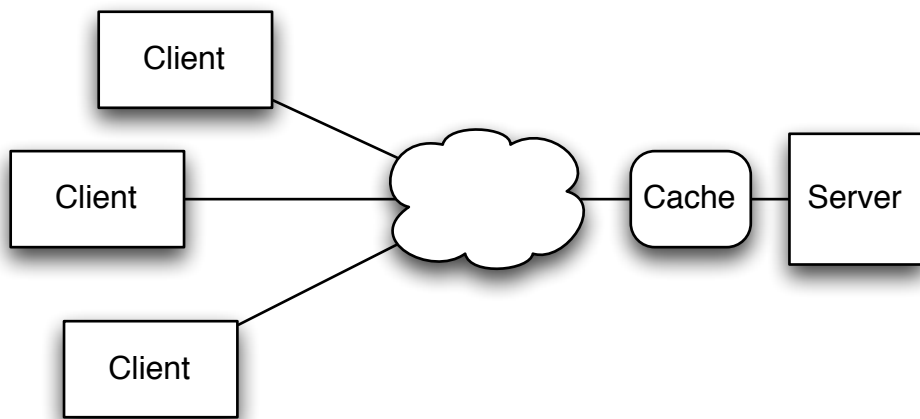
Cache Topologies



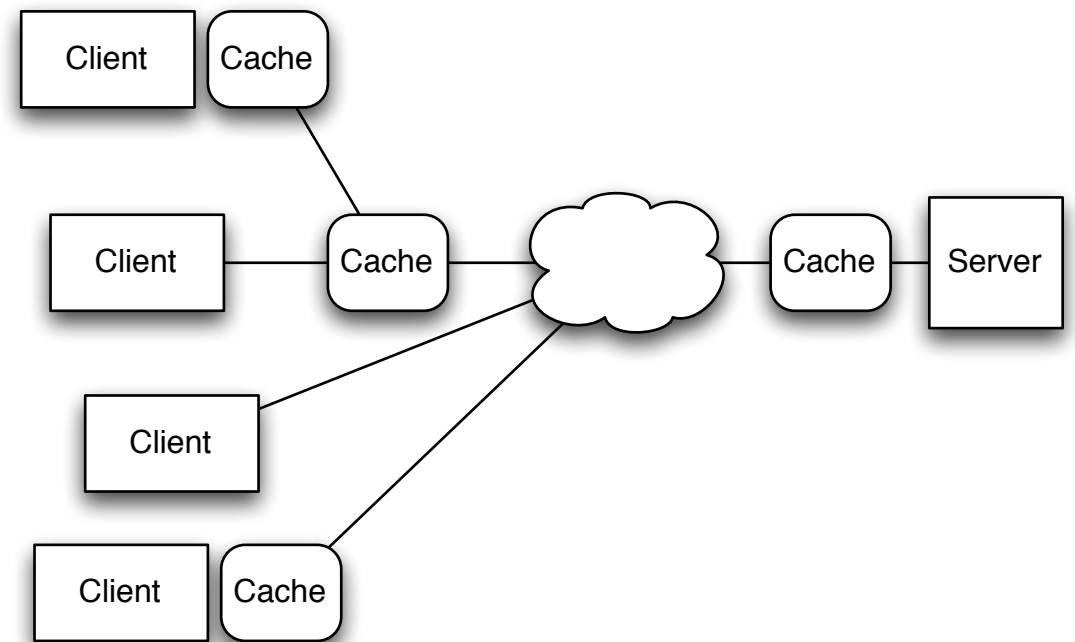
Client only



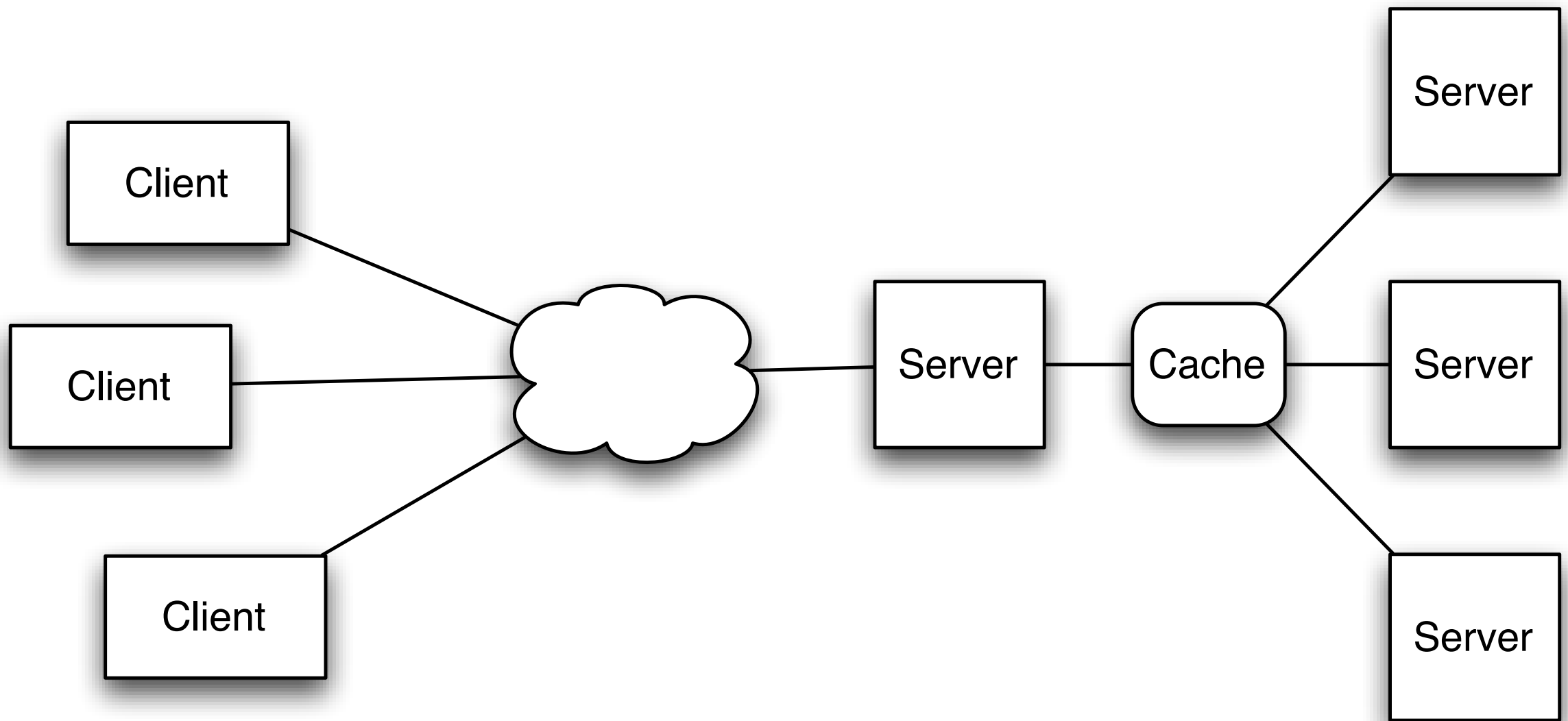
Proxy Cache



Reverse Proxy Cache



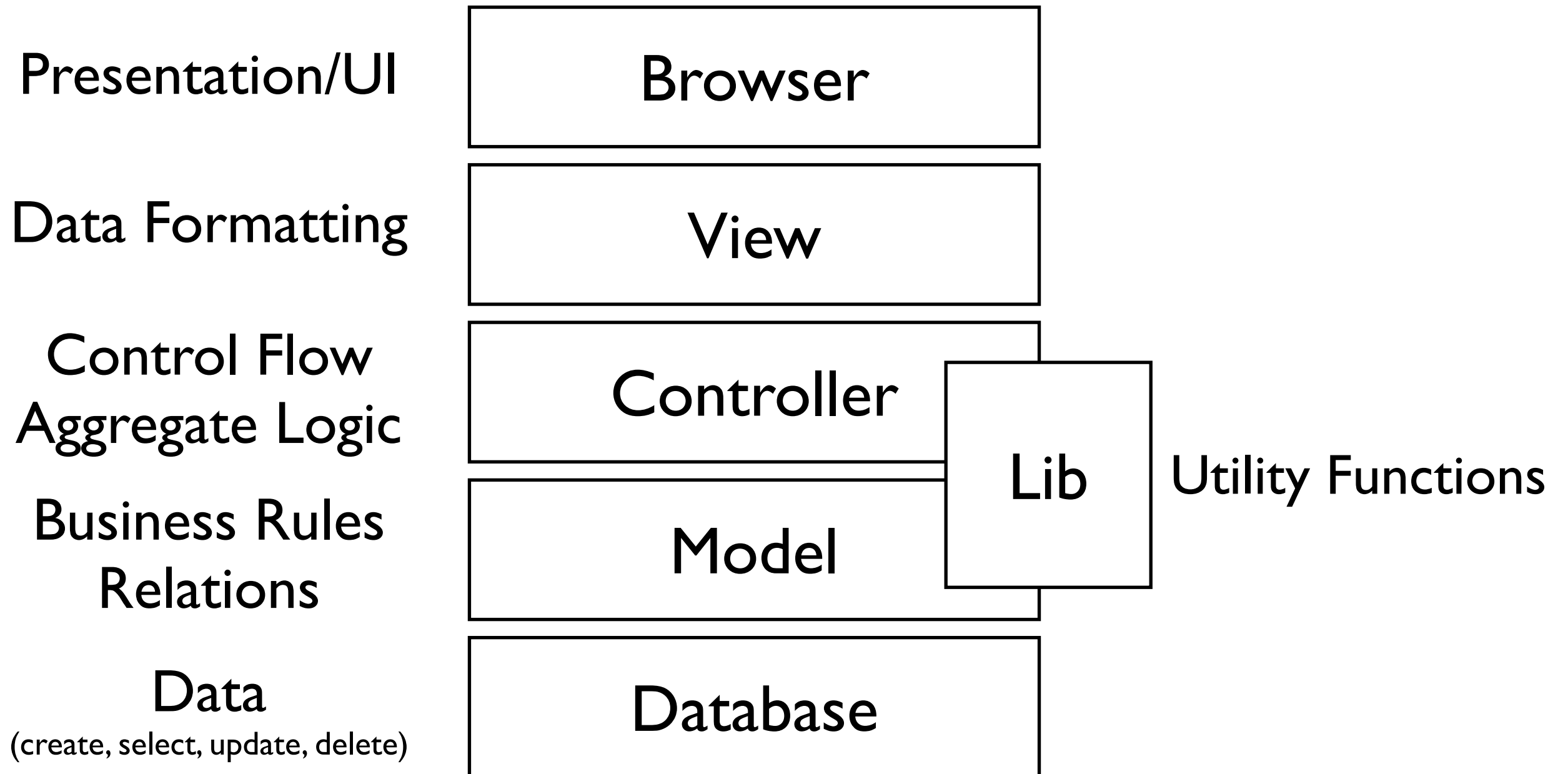
Complex Topology



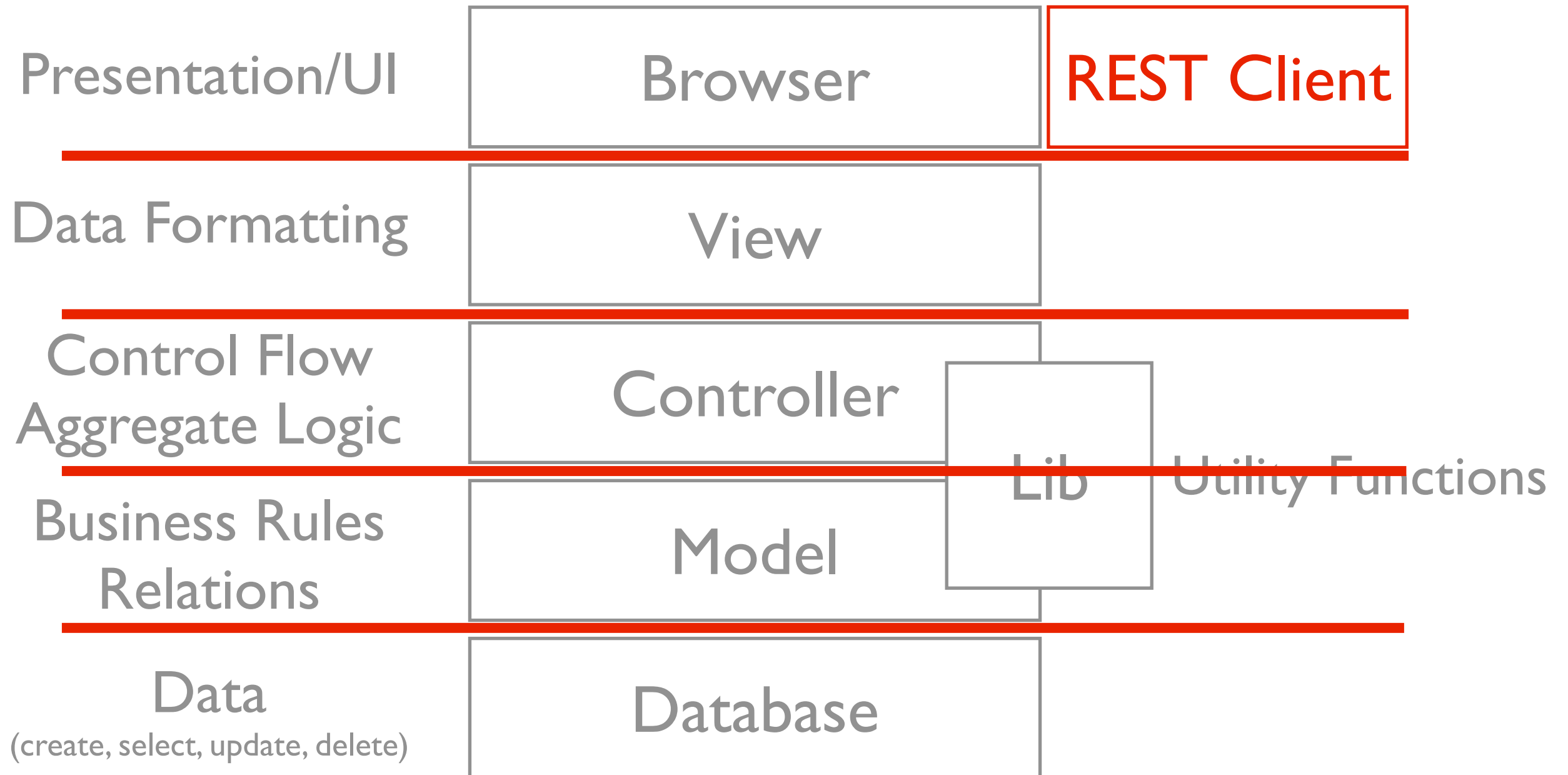
"Cache as ESB"

Web Sites vs. Web Services

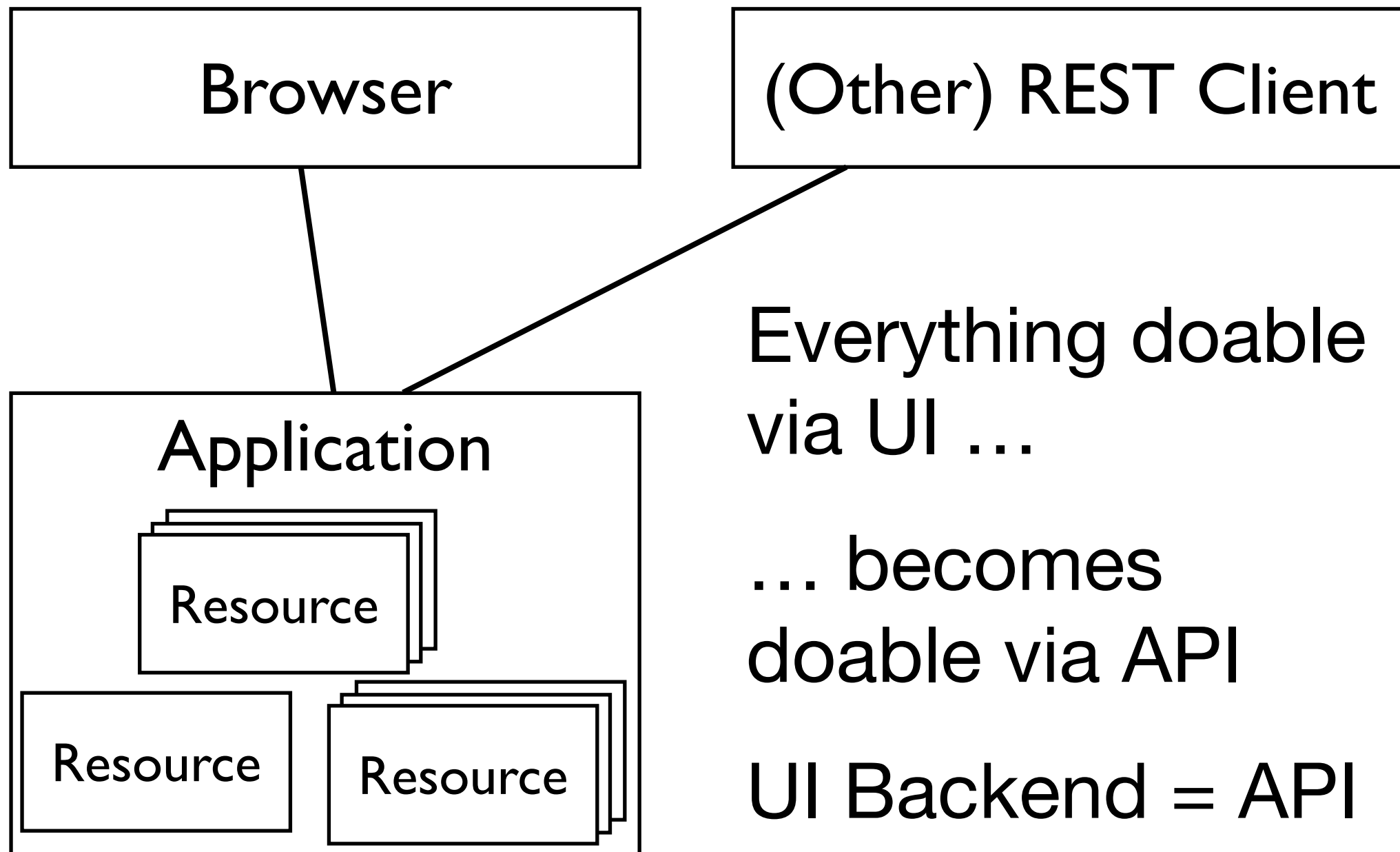
Application Layers



App Layers & Resources



Single Resource Model



Intermediaries

Squid

Full proxy cache w/ reverse proxy option

Mature/stable/old, widely used

Complicated configuration

Support for ESI

Support for external invalidation (PURGE)

(Experimental) support for cache channels

(much, much more)



mod_cache

mod_cache module for Apache HTTPD 2.x
(production-ready in 2.2)

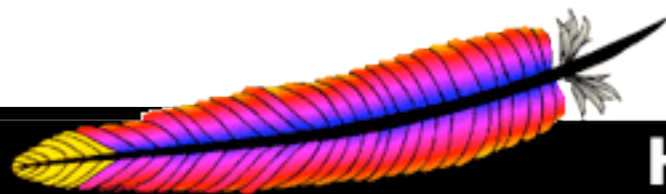
backends:

mod_mem_cache and **mod_disk_cache**

Runs within Apache process

Simple solution in conjunction with
Passenger

No support for ESI or explicit invalidation



Apache

HTTP SERVER PROJECT

HTTP СЕРВЕРЪ ПРОЕКТЪ

```
<VirtualHost 1.2.3.4>
  ServerName example.com
  <Location /images>
    ExpiresActive On
    ExpiresDefault A3600
  </Location>
  <Location /user>
    ExpiresActive On
    ExpiresDefault "access plus 1 month"
  </Location>
  CacheEnable disk /images
  CacheRoot /var/www/cache
  ProxyPass / http://localhost:3000
  ProxyPassReverse / http://localhost:3000
</VirtualHost>
```

Varnish

Pure in-memory reverse proxy

Disk storage through OS swap mechanism

(Partial) ESI support

External invalidation

VCL configuration language

Easy configuration



VCL

Hook	Meaning
vcl_recv	Called at the beginning of a request
vcl_pipe	Called upon entering pipe mode
vcl_pass	Called upon entering pass mode
vcl_hit	Called after a cache lookup if the requested document was found in the cache
vcl_miss	Called after a cache lookup if the requested document was not found in the cache
vcl_fetch	Called after a document has been successfully retrieved from the backend
vcl_deliver	Called before a cached object is delivered to the client
vcl_timeout	Called by the reaper thread shortly before a cached document reaches its expiry time
vcl_discard	Called by the reaper thread when a cached document is about to be discarded

```
sub vcl_recv {
  if (req.request != "GET" &&
      req.request != "HEAD" &&
      req.request != "PUT" &&
      req.request != "POST" &&
      req.request != "TRACE" &&
      req.request != "OPTIONS" &&
      req.request != "DELETE") {
    /* Non-RFC2616 or CONNECT
       which is weird. */
    return (pipe);
  }
  if (req.request != "GET"
      && req.request != "HEAD") {
    /* We only deal with GET and
       HEAD by default */
    return (pass);
  }
  if (req.http.Authorization
      || req.http.Cookie) {
    /* Not cacheable by default */
    return (pass);
  }
  return (lookup);
}
```

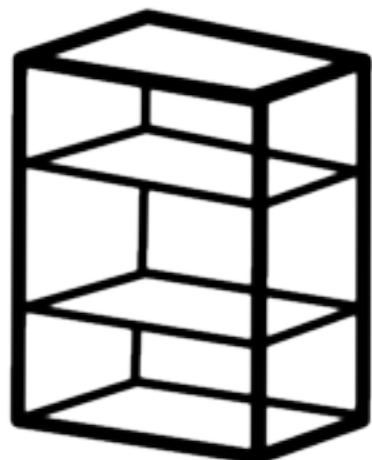
Rack::Cache

Runs within Ruby process

Cross-process synchronization via memcached

Absurdly simple + transparent

```
config.middleware.use(Rack::Cache,  
  :verbose => true,  
  :metastore => 'file:/var/cache/rack/meta',  
  :entitystore => 'file:/var/cache/rack/body')
```



rack::cache
powers web applications

Clients

Cache-aware HTTP Clients

Note: Conditional GET does not require anything special

Your browser and news reader

.NET (HttpWebRequest)

Python httplib2

HttpCache4J

So what?

How to Exploit HTTP Caching

1. Provide “safe” resource access via GET
2. Include appropriate validation and cache control headers
3. Leave caching to a reverse proxy cache
4. Use conditional GET on the client
5. Profit :-)

“I do think the REST-afarians are missing an opportunity by not driving home the secret sauce that is HTTP GET. [...] **GET is one of the most optimized pieces of distributed systems plumbing in the world.** It's an absolute/objective slam dunk. No arguing/evangelism needed IMO. GET is the classic ‘the first bag is free’ kind of feature a platform builder dreams about.”

Don Box, Co-inventor of SOAP

Thank you!
Any questions?

<http://www.innoq.com>

Stefan Tilkov

<http://www.innoq.com/blog/st/>



Architectural Consulting

SOA WS-* REST

MDA MDSD MDE

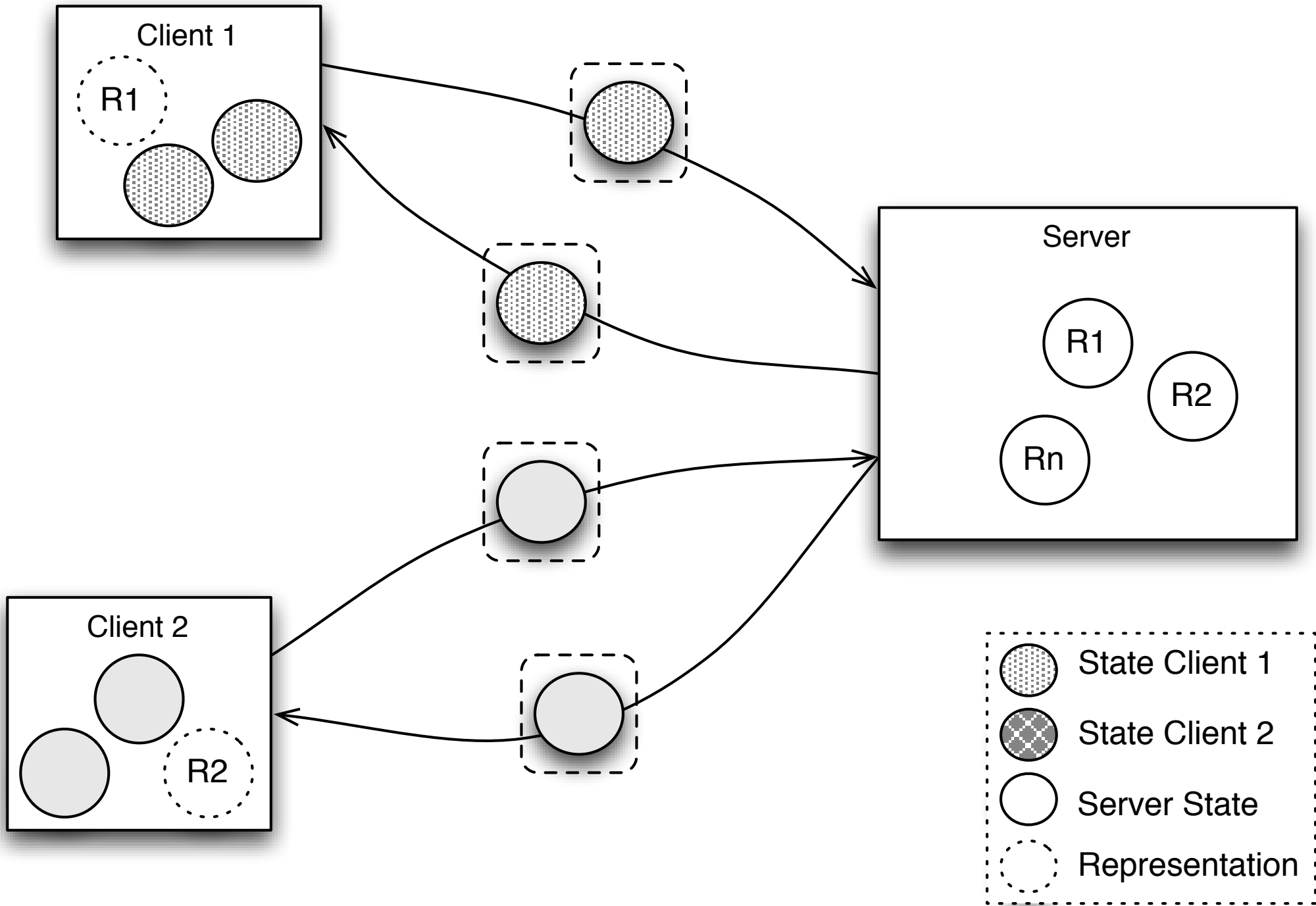
J(2)EE RoR .NET

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 2102 77 162-100
info@innoq.com · www.innoq.com

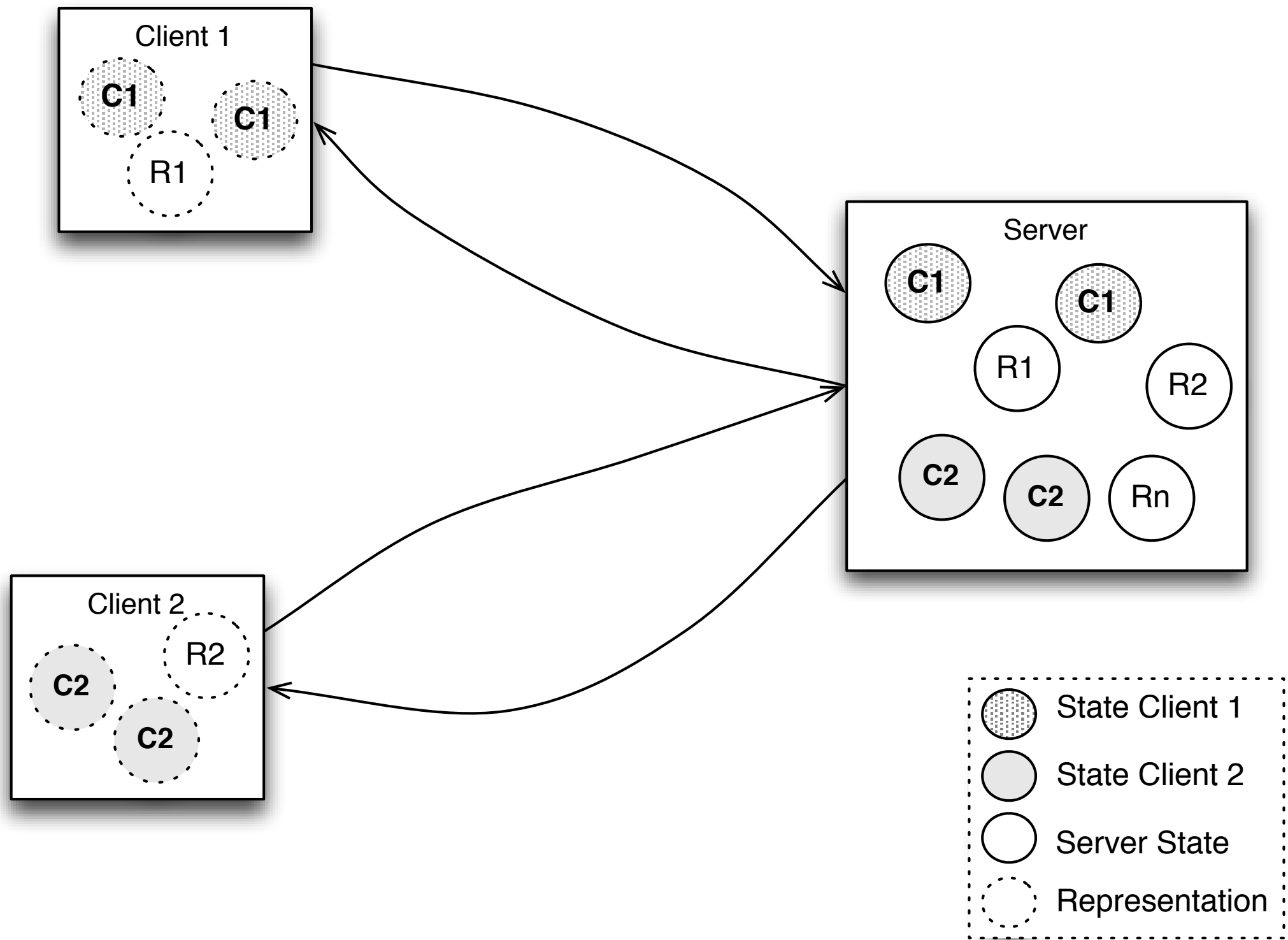
innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

Backup

Turning Session State ...



... into Resource or Client State



Resources

Nothingham, Mark: The State of Proxy Caching
http://www.mnot.net/blog/2007/06/20/proxy_caching

Nottingham, Mark: The State of Browser Caching
http://www.mnot.net/blog/2006/05/11/browser_caching

Nottingham, Mark: Caching Tutorial
http://www.mnot.net/cache_docs/

Squid, <http://www.squid-cache.org/>

Nottingham, Mark: Cache Channels for Squid
http://www.mnot.net/cache_channels/

The Varnish Project, <http://varnish.projects.linpro.no/>

Varnish: Notes from the Architect
<http://varnish.projects.linpro.no/wiki/ArchitectNotes>

Tomoyako, Ryan: Things Caches Do
<http://tomayko.com/writings/things-caches-do>

W3C, ESI Language Specification 1.0
<http://www.w3.org/TR/esi-lang>

Apache HTTP 2.2 Caching Guide
<http://httpd.apache.org/docs/2.2/caching.html>